



## PCI-SIG ENGINEERING CHANGE NOTICE

<b>TITLE:</b>	Trusted Configuration Space for PCI Express
<b>DATE:</b>	March 23, 2005, Updated 1 July 2005
<b>AFFECTED DOCUMENT:</b>	PCI Express Base Specification, Rev. 1.1
<b>SPONSOR:</b>	Microsoft Corporation, Intel Corporation

### **Part I**

#### **1. Summary of the Functional Changes**

10 This ECR proposes the addition of a new, optional address space, called Trusted Configuration Space (TCS). This new address space would allow software running in a Trusted Software Environment (TSE)<sup>1</sup> to configure and communicate with PCI Express devices by issuing Trusted Configuration Requests. These requests use new read and write Transaction Layer Packet (TLP) types that originate from the TSE. Access to TCS is provided via a new Trusted Configuration Access Mechanism (TCAM), which is intended to be resistant to snooping and interference by software running outside of the TSE on the same system.

#### **2. Benefits as a Result of the Changes**

15 Trusted Configuration Space enables the design of trusted computing platforms that provide assurance to Trusted Devices that they will only receive Trusted Configuration Requests from authorized software.

#### **3. Assessment of the Impact**

20 TCS support is optional for PCI Express devices. Devices that do not support TCS must follow the PCI Express Base Specification Rev. 1.1, Section 2.3, which describes the Transaction Layer behavior upon receipt of a Transaction Layer Packet (TLP) which uses an undefined (from the Device's perspective) Type field value. Additionally, device vendors must exercise special care when using generic interface cores in their implementations to ensure that their Device can not be re-purposed to masquerade as a Trusted Device that responds to Trusted Configuration Requests.

25 For PCI Express Switches and Root Ports, support for routing of Trusted Configuration Requests (also referred to as "TCS Routing") is optional, but strongly encouraged. Switches and Root Ports that do not support TCS routing must follow the Transaction Layer behavior described in PCI Express Base Specification Rev. 1.1, Section 2.3, upon receipt of a TLP that uses an undefined (from the Port's perspective) Type field value.

30 For PCI Express Root Complexes, support for TCS and a TCAM is optional. Systems providing a trusted computing environment must provide support for TCS and one or more TCAMs. Refer to Appendix X for more information on the characteristics and requirements of such an environment.

---

<sup>1</sup> A full description of a Trusted Software Environment is beyond the scope of this document. However, the requirements for a Trusted Software Environment, as they relate to TCS, are that the platform must at least provide a protected environment for running software that has a reliably-established notion of identity. This identity may be used by the platform manufacturer and/or platform owner to authorize a particular piece of software to access certain platform resources.

#### **4. Analysis of the Hardware Implications**

35 PCI Express Switches and Root Ports that support Trusted Configuration Space for downstream Trusted Devices are required to recognize new read and write Transaction Layer Packet (TLP) types and route them appropriately. These Switches and Root Ports must report their ability to properly route Trusted Configuration Requests by setting a "TCS Routing Supported" bit in the PCI Express Capabilities Register for both upstream and downstream ports.

40 PCI Express devices that provide a Trusted Configuration Space are required to recognize the new read and write TCS TLP types and provide Trusted Configuration Registers for configuration/communication via TCS. These Trusted Devices indicate that they are providing a Trusted Configuration Space by supplying a Configuration Access Correlation (CAC) extended capability in Standard Configuration Space. Once a Trusted Device has been thus identified, correlation is confirmed between Standard Configuration Space accesses and Trusted Configuration Space accesses by using the CAC extended capability in Standard Configuration Space in concert with a CAC Trusted Capability, which Trusted Devices are required to  
45 support in Trusted Configuration Space.

#### **5. Analysis of the Software Implications**

Accesses to any of the existing PCI Express address spaces (Memory, I/O, Configuration, Message) are unaffected by this change.

**Detailed Description of the change**

Change “Terms and Acronyms” section as follows:

**Terms and Acronyms**

...

Configuration Space	One of the <del>four</del> <u>five</u> address spaces within the PCI Express architecture. Packets with a <i>Configuration Space</i> address are used to configure a device.
I/O Space	One of the <del>four</del> <u>five</u> address spaces of the PCI Express architecture. Identical to the I/O space defined in PCI 3.0.
<u>Memory Space</u>	<u>One of the five address spaces of the PCI Express architecture. Identical to the Memory space defined in PCI 3.0. [editor note: added for completeness, not in 1.1]</u>
Message Space	One of the <del>four</del> <u>five</u> address spaces of the PCI Express architecture.
<u>Standard Configuration Space</u>	<u>See Configuration Space.</u>
<u>Trusted Device</u>	<u>A PCI Express Endpoint, Root Complex Integrated Endpoint, Root Port, or Switch that implements a Trusted Configuration Space as defined in this specification.</u>
<u>Trusted Configuration Access Mechanism, TCAM</u>	<u>An access mechanism available only on systems that support a Trusted Software Environment. This mechanism, accessible only from within the Trusted Software Environment, provides software with a way of accessing Trusted Configuration Space.</u>
<u>Trusted Configuration Space, TCS</u>	<u>One of the five address spaces of the PCI Express architecture. Used by systems having a Trusted Software Environment to communicate to PCI Express Trusted Devices without interference by software running outside of the Trusted Software Environment.</u>
<u>Trusted Software Environment, TSE</u>	<u>A protected environment for running software that has a reliably-established notion of identity. This identity is used by the platform manufacturer and/or platform owner to authorize a particular piece of software to access certain platform resources. (Refer to Appendix X for more details.)</u>

...

Change Sections 1 and 1.1 as follows:

**1. Introduction**

This chapter presents an overview of the PCI Express architecture and key concepts. PCI Express is a high performance, general purpose I/O interconnect defined for a wide variety of future computing and communication platforms. Key PCI attributes, such as its usage model, load-store architecture, and software interfaces, are maintained, whereas its parallel bus implementation is replaced by a highly scalable, fully serial interface. PCI Express takes advantage of recent advances in point-to-point interconnects, Switch-based technology, and packetized protocol to deliver new levels of performance

75 and features. Power Management, Quality Of Service (QoS), Hot-Plug/Hot-Swap support, Data Integrity, Trusted Configuration, and Error Handling are among some of the advanced features supported by PCI Express.

## 1.1 A Third Generation I/O Interconnect

The high-level requirements for this third generation I/O interconnect are as follows:

- 80 ...
- Ability to support differentiated services, i.e., different qualities of service (QoS)
  - ◆ Ability to have dedicated Link resources per QoS data flow to improve fabric efficiency and effective application-level performance in the face of head-of-line blocking
  - ◆ Ability to configure fabric QoS arbitration policies within every component
  - 85 ◆ Ability to tag end-to-end QoS with each packet
  - ◆ Ability to create end-to-end isochronous (time-based, injection rate control) solutions
  - Trusted Configuration support
  - ◆ Ability to support protected access to Trusted Configuration Space from a Trusted Software Environment.
  - 90 ◆ Ability to have trusted communication with Trusted Devices via Trusted Configuration Space
  - Hot-Plug and Hot-Swap support
  - ◆ Ability to support existing PCI Hot-Plug and Hot-Swap solutions
  - ◆ Ability to support native Hot-Plug and Hot-Swap solutions (no sideband signals required)
  - 95 ◆ Ability to support a unified software model for all form factors
- ...

*Change Section 1.3.1 as follows:*

### 1.3.1 Root Complex

- 100 ...
- ☐ A Root Complex must support generation of configuration requests as a Requester.
  - ☐ If the system supports a Trusted Software Environment, the Root Complex must support generation, from the TSE, of Trusted Configuration Requests as a Requester.
  - ☐ A Root Complex is permitted to support the generation of I/O requests as a Requester.
- 105 ...

Change Section 1.3.2.1 as follows:

### 1.3.2.1 Legacy Endpoint Rules

☐ A legacy Endpoint must be a device with a Type 00h Configuration Space header.

☐ If the legacy Endpoint is a Trusted Device:

- It must have a Trusted Configuration Space header.
- It must support Trusted Configuration Requests as a Completer.
- It must support a Configuration Access Correlation (CAC) extended capability in Standard Configuration Space and a CAC Trusted Capability in Trusted Configuration Space.

☐ A legacy Endpoint must support Configuration Requests as a Completer

☐ A legacy Endpoint may support I/O Requests as a Completer.

☐ A legacy Endpoint may generate I/O Requests.

☐ A legacy Endpoint may support Lock memory semantics as a Completer if that is required by the device's legacy software support requirements.

☐ A legacy Endpoint must not issue a Locked Request.

☐ A legacy Endpoint must not generate Trusted Configuration Requests.

☐ A legacy Endpoint may implement extended configuration space capabilities, but such capabilities may be ignored by software.

...

Change Section 1.3.2.2 as follows:

### 1.3.2.2 PCI Express Endpoint Rules

☐ A PCI Express Endpoint must be a device with a Type 00h Configuration Space header.

☐ If the PCI Express Endpoint is a Trusted Device:

- It must have a Trusted Configuration Space header.
- It must support Trusted Configuration Requests as a Completer.
- It must support a Configuration Access Correlation (CAC) extended capability in Standard Configuration Space and a CAC Trusted Capability in Trusted Configuration Space.

☐ A PCI Express Endpoint must support Configuration Requests as a Completer.

☐ A PCI Express Endpoint must not depend on operating system allocation of I/O resources claimed through BAR(s).

☐ A PCI Express Endpoint must not generate I/O Requests.

☐ A PCI Express Endpoint must not generate Trusted Configuration Requests.

☐ A PCI Express Endpoint must not support Locked Requests as a Completer or generate them as a Requestor. PCI Express-compliant software drivers and applications must be written to prevent the use of lock semantics when accessing a PCI Express Endpoint.

Change Section 1.3.2.3 as follows:

### 1.3.2.3 Root Complex Integrated Endpoint Rules

- ❑ A Root Complex Integrated Endpoint is implemented on internal logic of Root Complexes that contains the Root Ports.
- 145 ❑ A Root Complex Integrated Endpoint must be a device with a Type 00h Configuration Space header.
- ❑ If the Root Complex Integrated Endpoint is a Trusted Device:
  - It must have a Trusted Configuration Space header.
  - It must support Trusted Configuration Requests as a Completer.
  - 150 • It must support a Configuration Access Correlation (CAC) extended capability in Standard Configuration Space and a CAC Trusted Capability in Trusted Configuration Space.
- ❑ A Root Complex Integrated Endpoint must support Configuration Requests as a Completer
- ❑ A Root Complex Integrated Endpoint must not require I/O resources claimed through BAR(s).
- ❑ A Root Complex Integrated Endpoint must not generate I/O Requests.
- 155 ❑ A Root Complex Integrated Endpoint must not generate Trusted Configuration Requests.
- ❑ A Root Complex Integrated Endpoint must not support Locked Requests as a Completer or generate them as a Requestor. PCI Express-compliant software drivers and applications must be written to prevent the use of lock semantics when accessing a PCI Express Endpoint.

...

160 Change Section 1.5.1 as follows:

### 1.5.1 Transaction Layer

The upper Layer of the architecture is the Transaction Layer. The Transaction Layer's primary responsibility is the assembly and disassembly of Transaction Layer Packets (TLPs). TLPs are used to communicate transactions, such as read and write, as well as certain types of events. The Transaction Layer is also responsible for managing credit-based flow control for TLPs.

Every request packet requiring a response packet is implemented as a split transaction. Each packet has a unique identifier that enables response packets to be directed to the correct originator. The packet format supports different forms of addressing depending on the type of the transaction (Memory, I/O, Configuration, Trusted Configuration, and Message). The Packets may also have attributes such as No Snoop and Relaxed Ordering.

The transaction Layer supports ~~four~~ five address spaces: it includes the three PCI address spaces (memory, I/O, and configuration) and adds two more address spaces (Trusted Configuration and a Message) Space. This specification uses Message Space to support all prior sideband signals, such as interrupts, power-management requests, and so on, as in-band Message transactions. You could think of PCI Express Message transactions as "virtual wires" since their effect is to eliminate the wide array of sideband signals currently used in a platform implementation.

...

Change Section 2.1.1 as follows:

## 2.1.1 Address Spaces, Transaction Types, and Usage

- 180 | Transactions form the basis for information transfer between a Requester and Completer. ~~Four~~Five address spaces are defined within the PCI Express architecture, and different Transaction types are defined, each with its own unique intended usage, as shown in Table 2-1.

Table 2-1: Transaction Types for Different Address Spaces

Address Space	Transaction Types	Basic Usage
Memory	Read Write	Transfer data to/from a memory-mapped location.
I/O	Read Write	Transfer data to/from an I/O-mapped location
Configuration	Read Write	Device configuration/setup
<u>Trusted Configuration</u>	<u>Read</u> <u>Write</u>	<u>Trusted device configuration/setup</u>
Message	Baseline (including Vendor-defined)	From event signaling mechanism to general purpose messaging

- 185 | Details about the rules associated with usage of these address formats and the associated TLP formats are described later in this chapter.

...

Add Section 2.1.1.5 as follows:

### 2.1.1.5 Trusted Configuration Transactions

- 190 | Trusted Configuration Transactions are used to access trusted configuration registers of PCI Express Trusted Devices.

Trusted Configuration Transactions include the following types:

- ☐ Read Request/Completion
- ☐ Write Request/Completion

Change Section 2.2 as follows:

## 195 2.2 Transaction Layer Protocol - Packet Definition

PCI Express uses a packet based protocol to exchange information between the Transaction Layers of the two components communicating with each other over the Link. PCI Express supports the following basic transaction types: Memory, I/O, Configuration, Trusted Configuration, and Messages. Two addressing formats for Memory Requests are supported: 32 bit and 64 bit.





Add the following entries into Table 2-3, Section 2.2.1:

**Table 2-3: Fmt[1:0] and Type[4:0] Field Encodings**

TLP Type	Fmt [1:0] <sup>2</sup>	Type [4:0]	Description
MRd	00 01	0 0000	Memory Read Request
MRdLk	00 01	0 0001	Memory Read Request-Locked
MWr	10 11	0 0000	Memory Write Request
IORd	00	0 0010	I/O Read Request
IOWr	10	0 0010	I/O Write Request
CfgRd0	00	0 0100	Configuration Read Type 0
CfgWr0	10	0 0100	Configuration Write Type 0
CfgRd1	00	0 0101	Configuration Read Type 1
CfgWr1	10	0 0101	Configuration Write Type 1
<u>TCfgRd</u>	<u>00</u>	<u>11011</u>	<u>Trusted Configuration Read</u>
<u>TCfgWr</u>	<u>10</u>	<u>11011</u>	<u>Trusted Configuration Write</u>
Msg	01	1 0r <sub>2</sub> r <sub>1</sub> r <sub>0</sub>	Message Request – The sub-field r[2:0] specifies the Message routing mechanism (see Table 2-11).
MsgD	11	1 0r <sub>2</sub> r <sub>1</sub> r <sub>0</sub>	Message Request with data payload – The sub-field r[2:0] specifies the Message routing mechanism (see Table 2-11).
Cpl	00	0 1010	Completion without Data – Used for I/O <del>and</del> , Configuration <del>and</del> <u>Trusted Configuration</u> Write Completions and Read Completions (I/O, Configuration, <u>Trusted Configuration</u> or Memory) with Completion Status other than Successful Completion.
CplD	10	0 1010	Completion with Data – Used for Memory, I/O, <del>and</del> Configuration, <del>and</del> <u>Trusted Configuration</u> Read Completions.
CplLk	00	0 1011	Completion for Locked Memory Read without Data – Used only in error case.
CplDLk	10	0 1011	Completion for Locked Memory Read – otherwise like CplD.
			All encodings not shown above are Reserved.

<sup>2</sup> Requests with two Fmt[1:0] values shown can use either 32 bits (the first value) or 64 bits (the second value) Addressing Packet formats.

Change Section 2.2.4.2 as follows:

### **2.2.4.2 ID Based Routing Rules**

- 205 ☐ ID routing is used with Configuration Requests and Trusted Configuration Requests, optionally with Vendor\_Defined Messages (see Section 2.2.8.6), and with Completions

...

Change Section 2.2.5 as follows:

### **2.2.5 First/Last DW Byte Enables Rules**

- 210 | Byte Enables are included with Memory, I/O, ~~and~~ Configuration, and Trusted Configuration Requests. This section defines the corresponding rules. Byte Enables, when present in the Request header, are located in byte 7 of the header (see Figure 2-9).

...

Change Section 2.2.6.2 as follows:

### **2.2.6.2 Transaction Descriptor – Transaction ID Field**

...

- 220 ☐ Functions must capture the Bus and Device Numbers supplied with all Configuration Write Requests (Type 0) completed by the function and supply these numbers in the Bus and Device Number fields of the Requester ID for all Requests initiated by the device/function. Bus and Device Numbers must not be captured from Trusted Configuration Requests.

Exception: The assignment of Bus and Device Numbers to the logical devices within a Root Complex, and Device Numbers to the downstream ports within a switch, may be done in an implementation specific way.

- 225 Note that the Bus Number and Device Number may be changed at run time, and so it is necessary to re-capture this information with each and every Configuration Write Request.

- 230 Example: When a device (or function of a multi-function device) receives a Type 0 Configuration Write Request, the device comprehends that it is the intended recipient of the Request because it is a Type 0 Request. The routing information fields of the Write Request include the recipient's Bus Number and Device Number values (Figure 2-16). These values are captured by the device and used to generate the Requester and Completer ID field.

...

Change Section 2.2.7 as follows:

### **2.2.7 Memory, I/O, ~~and~~ Configuration, and Trusted Configuration Request Rules**

- 235 The following rule applies to all Memory, I/O ~~and~~ Configuration and Trusted Configuration Requests. Additional rules specific to each type of Request follow.

- ☐ All Memory, I/O, ~~and~~ Configuration and Trusted Configuration Requests include the following fields in addition to the common header fields:

240

- Requester ID[15:0] and Tag[7:0], forming the Transaction ID
- Last DW BE[3:0] and 1st DW BE[3:0]

...

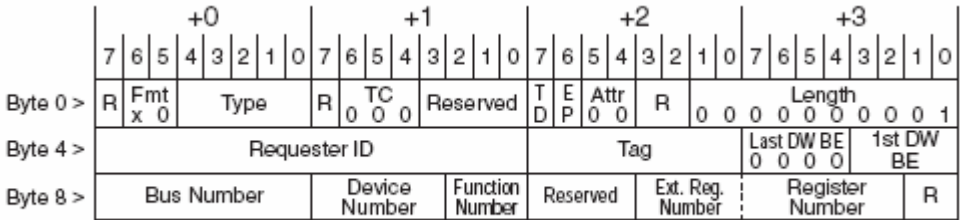
For Configuration and Trusted Configuration Requests, the following rules apply:

245

- ❑ Configuration and Trusted Configuration Requests route by ID, and use a 3 DW header
- ❑ In addition to the header fields included in all Memory, I/O, ~~and~~ Configuration and Trusted Configuration Requests and the ID routing fields, Configuration and Trusted Configuration Requests contain the following additional fields (see Figure 2-16)
- Register Number[5:0]
- Extended Register Number[3:0]

250

- ❑ Configuration and Trusted Configuration Requests have the following restrictions:
- TC[2:0] must be 000b
- Attr[1:0] must be 00b
- Length[9:0] must be 00 0000 0001b
- Last DW BE[3:0] must be 0000b



OM13766

**Figure 2-16: Request Header Format for Configuration and Trusted Configuration Transactions**

255

...

Change Section 2.2.9 as follows:

### **2.2.9 Completion Rules**

...

260

- ❑ Functions must capture the Bus and Device Numbers supplied with all Type 0 Configuration Write Requests completed by the function, and supply these numbers in the Bus and Device Number fields of the Completer ID for all Completions generated by the device/function. Bus and Device Numbers must not be captured from Trusted Configuration Requests.
- If a function must generate a Completion prior to the initial device Configuration Write Request, 0's must be entered into the Bus Number and Device Number fields
- Note that Bus Number and Device Number may be changed at run time, and so it is necessary to re-capture this information with each and every Configuration Write Request.

265

- Exception: The assignment of bus numbers to the logical devices within a Root Complex may be done in an implementation specific way.

270    ☐ A Trusted Device is not permitted to return a Completion with Successful status until the Device's Requester ID has been captured from a Standard Configuration Write Request to the Device. Trusted Configuration Requests received prior to this time must be completed with UR status, and 0's must be entered into the Bus Number and Device Number fields of the Completer ID.

...

*Change Section 2.3.1 as follows:*

275    ...

- For Configuration Requests only (not Trusted Configuration Requests), following reset it is possible ...

*Change Section 2.3.1.1 as follows:*

### 280    **2.3.1.1 Data Return for Read Requests**

- ☐ Individual Completions for Memory Read Requests may provide less than the full amount of data Requested so long as all Completions for a given Request when combined return exactly the amount of data Requested in the Read Request.
  - Completions for different Requests cannot be combined.
  - 285    • I/O ~~and~~ Configuration and Trusted Configuration Reads must be completed with exactly one Completion.
  - The Completion Status for a Completion corresponds only to the status associated with the data returned with that Completion
  - 290    ♦ A Completion with status other than Successful Completion terminates the Completions for a single Read Request
    - In this case, the value in the Length field is undefined, and must be ignored by the Receiver

...

*Change Section 2.4.1 as follows:*

### 295    **2.4.1 Transaction Ordering Rules**

Table 2-23 defines the ordering requirements for PCI Express Transactions. The rules defined in this table apply uniformly to all types of Transactions on PCI Express including Memory, I/O, Configuration, Trusted Configuration, and Messages. The ordering rules defined in this table apply within a single Traffic Class (TC). There is no ordering requirement among transactions with different TC labels. Note that this also implies that there is no ordering required between traffic that flows through different Virtual Channels since transactions with the same TC label are not allowed to be mapped to multiple VCs on any PCI Express Link.

300

...

**Table 2-23: Ordering Rules Summary Table**

Row Pass Column?		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Col 2)	Read Request (Col 3)	I/O- <del>or</del> Configuration or Trusted Configuration Write Request (Col 4)	Read Completion (Col 5)	I/O- <del>or</del> Configuration or Trusted Configuration Write Completion (Col 6)
Posted Request	Memory Write or Message Request (Row A)	a) No b) Y/N	Yes	Yes	a) Y/N b) Yes	a) Y/N b) Yes
	Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
Non-Posted Request	I/O- <del>or</del> Configuration or Trusted Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
	Read Completion (Row D)	a) No b) Y/N	Yes	Yes	a) Y/N b) No	Y/N
Completion	I/O- <del>or</del> Configuration or Trusted Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

305 Explanation of the entries in Table 2-23:

A2 a A Memory Write or Message Request with the Relaxed Ordering Attribute bit clear (0b) must not pass any other Memory Write or Message Request.

A2 b A Memory Write or Message Request with the Relaxed Ordering Attribute bit set (1b) is permitted to pass any other Memory Write or Message Request.

310 A3, A4 A Memory Write or Message Request must be allowed to pass Read Requests and I/O-~~or~~ Configuration, or Trusted Configuration Write Requests to avoid deadlocks.

A5, A6 a Endpoints, Switches, and Root Complex may allow Memory Write and Message Requests to pass Completions or be blocked by Completions.

315 A5, A6 b PCI Express to PCI Bridges and PCI Express to PCI-X Bridges, when operating PCI segment in conventional mode, must allow Memory Write and Message Requests to pass Completions traveling in the PCI Express to PCI direction (Primary side of Bridge to Secondary side of Bridge) to avoid deadlock.

B2, C2 These Requests cannot pass a Memory Write or Message Request. This preserves strong write ordering required to support Producer/Consumer usage model.

320	B3, B4, C3, C4	Read Requests and I/O- <del>or</del> <sub>3</sub> Configuration <u>or Trusted Configuration</u> Write Requests are permitted to be blocked by or to pass other Read Requests and I/O- <del>or</del> <sub>3</sub> Configuration <u>or Trusted Configuration</u> Write Requests.
325	B5, B6, C5, C6	These Requests are permitted to be blocked by or to pass Completions.
	D2 a	If the Relaxed Ordering attribute bit is not set, then a Read Completion cannot pass a previously enqueued Memory Write or Message Request.
	D2 b	If the Relaxed Ordering attribute bit is set, then a Read Completion is permitted to pass a previously enqueued Memory Write or Message Request.
330	D3, D4, E3, E4	Completions must be allowed to pass Read and I/O- <del>or</del> <sub>3</sub> Configuration <u>or Trusted Configuration</u> Write Requests to avoid deadlocks.
	D5 a	Read Completions associated with different Read Requests are allowed to be blocked by or to pass each other.
335	D5 b	Read Completions for one Request (will have the same Transaction ID) must return in address order.
	D6	Read Completions are permitted to be blocked by or to pass I/O- <del>or</del> <sub>3</sub> Configuration <u>or Trusted Configuration</u> —Write Completions.
340	E2	I/O- <del>or</del> <sub>3</sub> Configuration <u>or Trusted Configuration</u> Write Completions are permitted to be blocked by or to pass Memory Write and Message Requests. Such Transactions are actually moving in the opposite direction and, therefore, have no ordering relationship.
345	E5, E6	I/O- <del>or</del> <sub>3</sub> Configuration <u>or Trusted Configuration</u> Write Completions are permitted to be blocked by or to pass Read Completions and other I/O- <del>or</del> <sub>3</sub> Configuration <u>or Trusted Configuration</u> Write Completions.
	...	

Change Section 2.7.2.2 as follows:

### 2.7.2.2. Rules For Use of Data Poisoning

- ☐ Support for TLP poisoning in a Transmitter is optional.
- 350 ☐ Data poisoning applies only to the data within a Write Request (Posted or Non-Posted) or a Read Completion.
  - Poisoning of a TLP is indicated by a 1b value in the EP field
- ☐ If a Transmitter supports data poisoning, TLPs that are known to the Transmitter to include bad data must use the poisoning mechanism defined above.
- 355 ☐ Receipt of a Poisoned TLP is a reported error associated with the Receiving device/function (see Section 6.2)

- ❑ A Poisoned Configuration Write Request or Trusted Configuration Write Request must be discarded by the Completer, and a Completion with a Completion Status of UR is returned (see Section 2.2.9).

- A Switch must route a Poisoned Configuration Write Request or Trusted Configuration Write Request in the same way it would route a non-Poisoned ~~Configuration Write~~ Request, unless the Request addresses the configuration space of the Switch itself, in which case the Switch is the Completer for the Request and must follow the above rule.

...

365 Add Sections 6.X and 6.X.1 as follows:

## **6.X Configuration For Trusted Software Environments**

The following PCI Express features provide support for a Trusted Software Environment (TSE).

- ❑ A new address space: “Trusted Configuration Space”
- ❑ Transactions to/from new address space: “Trusted Configuration Requests”
- ❑ A new access mechanism: “Trusted Configuration Access Mechanism”

The Trusted Configuration Access Mechanism must be usable only from within a TSE.

### **6.X.1 Trusted Configuration Space**

Trusted Configuration Space (TCS) is used to communicate from a Trusted Software Environment running on the system to PCI Express Trusted Devices. TCS is not designed to prevent hardware based attacks where test equipment or modified hardware is used. Access to TCS from within a Trusted Software Environment is provided by the Trusted Configuration Access Mechanism.

While TCS address space is independent of Standard Configuration address space, it is permitted for a Trusted Device to have certain registers that map into both address spaces. This mapping to both address spaces is covered on a per register basis in the Trusted Device register definition section of this document. PCI Express Switches that support routing of Trusted Configuration Requests (also referred to as “TCS routing”) must only route these Trusted Configuration Requests downstream and must not propagate upstream any Trusted Configuration Requests arriving at one of their downstream ports. This is the same behavior as for Configuration Requests.

Only the root complex is permitted to initiate Trusted Configuration Requests and only in response to a request from a Trusted Software Environment. No other PCI Express entity is permitted to initiate a Trusted Configuration Request.

Support of Trusted Configuration Space is optional for PCI Express devices, and support for routing of Trusted Configuration Requests is optional for Root Ports and Switches. An upstream Trusted Configuration Request received at a downstream port is a Malformed TLP. A downstream Trusted Configuration Request received at the upstream port of a Switch that does not support TCS routing is a Malformed TLP. Software must not attempt to transmit a Trusted Configuration Request to any device that does not indicate support for Trusted Configuration Space or through any Switch port that does not indicate support for TCS routing. Software must not attempt to transmit a Trusted Configuration Request through a Root Port that indicates a lack of support for TCS routing, and the behavior of the Root Complex in this case is undefined.

PCI Express Switches must not modify Trusted Configuration Requests<sup>3</sup>, nor are they permitted to convert other types of requests to Trusted Configuration Requests. This also applies to debug modes of the Switch. PCI Express Root Complexes implement Trusted Configuration Space only on platforms that provide a Trusted Software Environment, as discussed in Appendix X.

400 *Change Section 7.2.2.1 as follows:*

### **7.2.2.1 Host Bridge Requirements**

405 For those systems that implement the enhanced configuration access mechanism, the PCI Express Host Bridge is required to translate the memory-mapped PCI Express configuration space accesses from the host processor to PCI Express configuration transactions. The use of Host Bridge PCI class code is reserved for backwards compatibility; host Bridge configuration space is opaque to standard PCI Express software and may be implemented in an implementation specific manner that is compatible with PCI Host Bridge Type 0 configuration space. A PCI Express Host Bridge is not required to signal errors through a Root Complex Event Collector. This support is optional for PCI Express Host Bridges.

410 A PCI Express Host Bridge that supports the Trusted Configuration Access Mechanism must translate memory-mapped host processor accesses into Trusted Configuration transactions, per the rules outlined in Section 7.2.4.1.

*Add Sections 7.2.4, 7.2.4.1, and 7.2.4.2 as follows:*

## **7.2.4 Trusted Configuration Space**

415 Trusted Configuration Space (TCS) is an optional additional PCI Express address space that allows software running in a Trusted Software Environment (TSE) to configure and communicate with PCI Express Trusted Devices by issuing Trusted Configuration Requests. These requests use read and write Transaction Layer Packet (TLP) types that only originate from a TSE. Access to TCS is provided via a Trusted Configuration Access Mechanism.

420 Trusted Configuration Space is 4 KB in size and consists of a fixed size header (starting at offset 000h) followed by a linked list of variable sized capabilities. The header contains the offset of the first Trusted Capability. Trusted Configuration requests are routed by ID (bus number, device number, function number).

### **7.2.4.1 Trusted Configuration Access Mechanism**

425 A PCIe Trusted Configuration Access Mechanism (TCAM) provides a flat memory-mapped address range to generate Trusted Configuration Requests. A TCAM closely parallels the PCIe Enhanced Configuration Access Mechanism (ECAM) described in Section 7.2.2, only it uses a different Base Address and is usable only by software running within a TSE.

430 For systems that provide a processor-architecture-specific firmware interface for accessing Normal Configuration Space instead of providing an ECAM, it is permitted for those systems to provide a

---

<sup>3</sup> When necessary, Switches are permitted to poison a Trusted Configuration Write Request by setting the EP field value to 1b. See Section 2.7.2.2.



firmware interface for accessing Trusted Configuration Space instead of providing a TCAM. In such implementations, the firmware must support execution from within the TSE.

A system that exposes a TCAM to an operating system must obey the following rules:

- ☐ The size of the memory address range for the TCAM must be the same as the memory address range for the corresponding ECAM.
- ☐ The mapping from memory address space to PCIe Trusted Configuration space address (Bus Number, Device Number, Function Number, etc) parallels the mappings for an ECAM, as defined in Table 7-1.
- ☐ The size and Base Address of the TCAM are reported by the firmware to the operating system in an implementation-specific manner.
- ☐ A TCAM is not required to support accesses larger than a DW, or accesses that cross a DW boundary.
- ☐ A TCAM is not required to support a locked access.
- ☐ System hardware must provide a method for system software to guarantee that a write transaction using the TCAM is completed by the completer before system software execution continues.

In all systems, device drivers are encouraged to use the application programming interface (API) provided by the operating system to access the Configuration Space of its device and not directly use the hardware mechanism.



## **IMPLEMENTATION NOTE**

### **Generating Configuration Requests via the TCAM**

Because Root Complex implementations are not required to support the generation of Trusted Configuration Requests from memory space accesses that cross DW boundaries, or that use locked semantics, software should take care not to cause the generation of such requests when using the TCAM unless it is known that the Root Complex implementation being used supports the translation.

## **7.2.4.2 PCI Express Trusted Device Requirements**

Trusted Devices must support Trusted Configuration Requests and must correctly decode the full 4KB of device Trusted Configuration Space (without aliasing), i.e. decode the Extended Register Address[3:0] field of the Trusted Configuration Request header in addition to the Bus, Device, Function and Register number fields.

*Add Sections 7.3.5 and 7.3.6, after Section 7.3.4, as follows:*

## **7.3.5 Trusted Configuration Transaction Addressing**

PCI Express Trusted Configuration Requests use the following addressing fields:

- ☐ Bus Number – PCI Express maps logical PCI Bus Numbers onto PCI Express Links such that PCI 3.0 compatible configuration software views the Configuration Space of a PCI Express Hierarchy as a PCI hierarchy including multiple bus segments. Same as Standard Configuration Space.

- ❑ Device Number – Device Number association is discussed in Section 7.3.1. Same as Standard Configuration Space.
- ❑ Function Number – PCI Express also supports multi-function devices using the same discovery mechanism as PCI 3.0. Same as Standard Configuration Space. Note that all PCI Express Trusted Devices (including single function devices) must decode the Function Number.
- ❑ Extended Register Number and Register Number – Specify the Trusted Configuration Space address of the register being accessed (concatenated such that the Extended Register Number forms the more significant bits). Trusted Configuration Space at the device level is independent of Configuration Space.

### **7.3.6 Trusted Configuration Request Routing Rules**


For PCI Express Endpoint and Root Complex Integrated Endpoint devices, the following rules apply:

- ❑ If a Trusted Configuration Request and a Trusted Device,
  - Determine if the Request addresses a valid local Trusted Configuration Space
    - ◆ If so, process the Request
    - ◆ If not, follow rules for handling Unsupported Requests

- ❑ If a Trusted Configuration Request and not a Trusted Device,
  - Follow rules for handling Malformed Packets

For Root Ports, Switches, and PCI Express-PCI/PCI-X Bridges, the following rules apply:

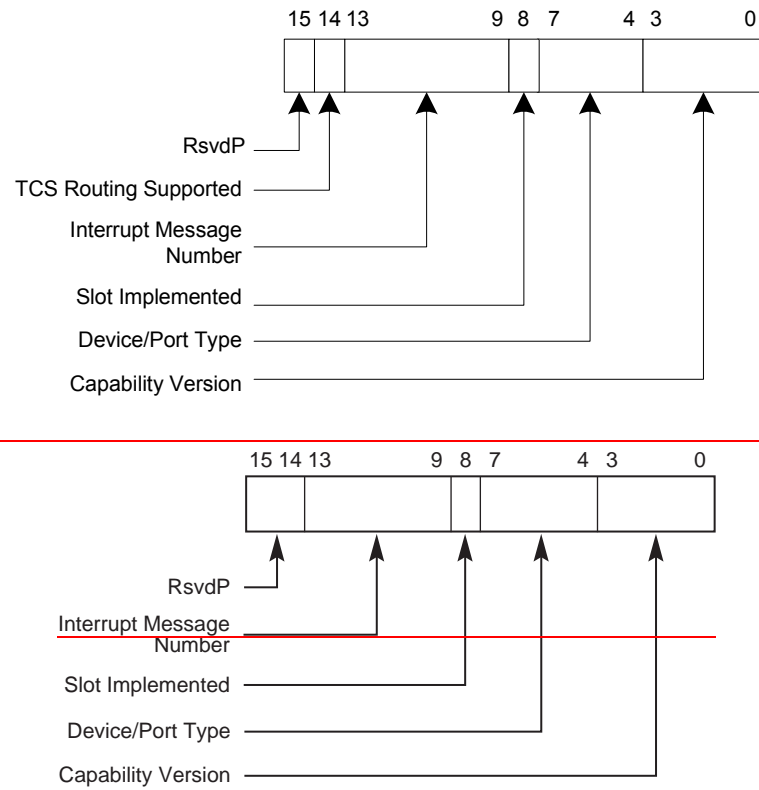
- ❑ Propagation of Trusted Configuration Requests from Downstream to Upstream as well as peer-to-peer are not supported
  - Trusted Configuration Requests are initiated only by the Host Bridge
- ❑ If a Trusted Configuration Request and this is a Switch or Root Port that supports routing of Trusted Configuration Requests:
  - Determine if the Request addresses a valid local trusted configuration space
    - ◆ If so, process the Request
    - ◆ If not, apply the following tests to the Bus Number field:
      - If, in the case of a Switch or Root Port, equal to the bus number and decoded device numbers assigned to one of the Root Port or Downstream Ports of the Switch, or in the range of bus numbers assigned to one of a Downstream Port
        - Forward the Request to that Downstream Port interface without modification
      - Else
        - The Request is invalid - follow the rules for handling Unsupported Requests
- ❑ If a Trusted Configuration Request and this is a Root Port or Switch that does not support routing of Trusted Configuration Requests:
  - If a Trusted Device,
    - ◆ Determine if the Request addresses a valid local trusted configuration space

	<ul style="list-style-type: none"> <li>○ <u>If so, process the Request</u></li> <li>○ <u>If not, follow rules for handling Unsupported Requests</u></li> </ul>
505	<ul style="list-style-type: none"> <li>• <u>Else</u> <ul style="list-style-type: none"> <li>◆ <u>The Request is a malformed packet – discard and report Malformed Packet</u>  <u>Note: This type of access is a consequence of a programming error.</u></li> </ul> </li> <li>❑ <u>PCI Express-PCI Bridges must treat any Trusted Configuration Requests directed towards a PCI bus as Malformed Packets. The packet must be discarded and Malformed Packet error reported.</u>  <u>Note: This type of access is a consequence of a programming error.</u></li> </ul>
510	<u>Additional rule specific to Trusted Configuration Transactions:</u>
	<ul style="list-style-type: none"> <li>❑ <u>Trusted Configuration Transactions must not be used to set up the PCI Express hierarchy either for baseline configuration or for hierarchy reconfiguration. Regular PCI Express configuration transactions must be used for these purposes.</u></li> </ul>
515	<ul style="list-style-type: none"> <li>❑ <u>Trusted Configuration Transactions must not be used before Standard Configuration Requests have completed assignment of Bus Numbers and established hierarchy for the PCI Express interconnect.</u></li> <li>❑ <u>A Trusted Device must not respond with "Configuration Request Retry Status" to a Trusted Config Request.</u></li> </ul>
520	<div>  <h2><u>IMPLEMENTATION NOTE</u></h2> <h3><u>Accessing Trusted Configuration Space</u></h3> <p><u>Software must complete initial PCI enumeration before attempting to access the TCS of a Trusted Device.</u></p> </div>

*Modify Section 7.8.2 as follows:*

### **7.8.2. PCI Express Capabilities Register (Offset 02h)**

525 The PCI Express Capabilities register identifies PCI Express device type and associated capabilities. Figure 7-11 details allocation of register fields in the PCI Express Capabilities register; Table 7-10 provides the respective bit definitions.



OM14502

**Figure 7-11: PCI Express Capabilities Register**

Table 7-10: PCI Express Capabilities Register

Bit Location	Register Description	Attributes
...		
<b>14</b>	<p><b><u>TCS Routing Supported</u></b> – This bit when set indicates that the PCI Express Switch or Root Port supports routing of Trusted Configuration Requests. Switches that support routing of Trusted Configuration Requests must set this bit on both upstream and downstream ports. Root Ports that support routing of Trusted Configuration Requests must also set this bit.</p> <p><u>It is permitted for a Root Port to set this bit even if the TCAM is not available on the platform, in which case the Root Complex as a whole does not support generation of Trusted Configuration Requests.</u></p> <p><u>Implementing the TCS Routing Supported capability bit in standard PCI Express Configuration Space allows software to determine whether a given Trusted Device's TCS may be accessed, based on whether or not the Root Port and all intervening Switches properly route Trusted Configuration Requests.</u></p> <p><u>This mechanism is useful, for example, to facilitate troubleshooting in systems where the user has plugged a Trusted Device into a slot that is downstream of a switch that doesn't support TCS routing. Software could detect this condition, and assist the user in finding a suitable slot in which to plug in their Trusted Device.</u></p>	<b><u>RO</u></b>



## **IMPLEMENTATION NOTE**

### **Identifying Physical Slots That Accept Trusted Devices**

In order to facilitate user placement and troubleshooting of Trusted add-in cards, it is recommended that system OEMs support a Physical Slot Identifier (as defined in PCI HP 1.1, Section 1.5) for each slot on the system board, with user-visible labels clearly associating each physical slot with its corresponding slot number.

Add Sections 7.20, 7.20.1, and 7.20.2 as follows:

## **7.20 Configuration Access Correlation Capability**

The PCI Express Configuration Access Correlation (CAC) capability is an optional extended capability that must be implemented by any PCI Express device that provides a Trusted Configuration Space. The CAC is used by software running within the TSE to identify Trusted Devices. Contained within the CAC is a read-only 32-bit value that is used by software running within the TSE to validate that Standard Configuration Space accesses performed via ECAM are targeting the same device as Trusted Configuration Space accesses performed via the corresponding TCAM. Figure 7-87 details allocation of register fields in the PCI Express Capability structure.

545 PCI Express devices that implement this capability must also implement the corresponding CAC  
Trusted Capability in Trusted Configuration Space (see Section 7.21.2). Both capabilities must access  
the same register for the Device Correlation—it is by this means that correlation is verified by software  
running in the TSE.

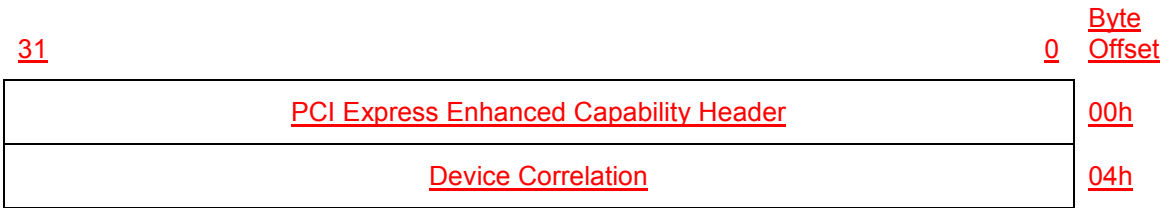
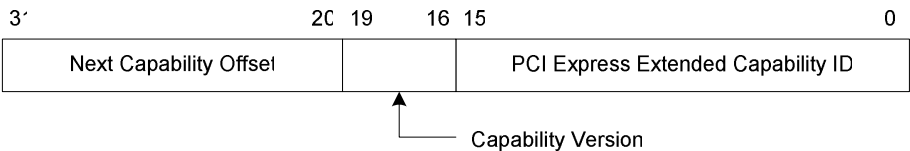


Figure 7-87: Configuration Access Correlation Capability

550 **7.20.1 Configuration Access Correlation Capability Header (Offset 00h)**

Figure 7-88 details allocation of register fields in the CAC Enhanced Capability header; Table 7-75 provides the respective bit definitions. The Extended Capability ID for the CAC Capability is 000Ch.



555 **Figure 7-88: Configuration Access Correlation Capability Header**

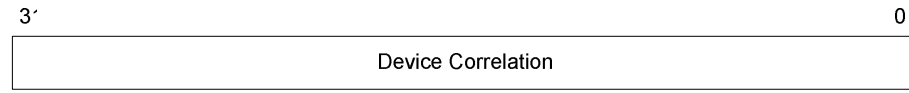
**Table 7-75: Trusted Configuration Space Capability Header**

Bit Location	Register Description	Attributes
15:0	<b>PCI Express Extended Capability ID</b> – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability.  Extended Capability ID for the Configuration Access Correlation Capability is 000Ch.	RO
19:16	<b>Capability Version</b> – This field is a PCI-SIG defined version number that indicates the version of the capability structure present.  Must be 1h for this version of the specification.	RO
31:20	<b>Next Capability Offset</b> – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of capabilities.  For Extended Capabilities implemented in device Configuration Space, this offset is relative to the beginning of PCI compatible Configuration Space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

	The bottom two bits of this offset are reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.	
--	---	--

## 7.20.2 Device Correlation Register (Offset 04h)

The Device Correlation register is a 32-bit read-only register that is used by software running within the TSE to confirm that Standard Configuration Requests and Trusted Configuration Requests for a given bus, device, and function are targeting the same device. Figure 7-89 details allocation of register fields in the Device Correlation register; Table 7-76 provides the respective bit definitions.



**Figure 7-89: Device Correlation Register**

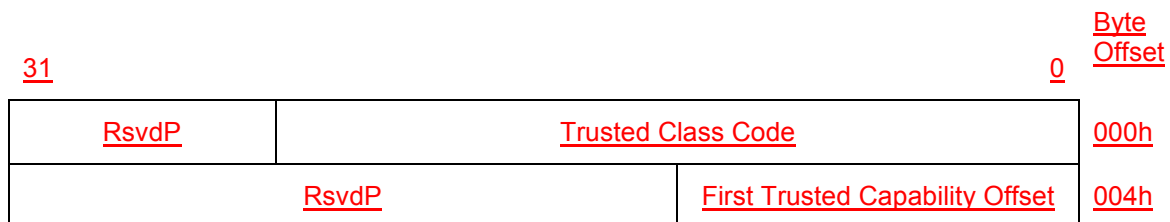
**Table 7-76: Device Correlation Register**

Bit Location	Register Description	Attributes
31:0	<b>Device Correlation</b> – This field contains a value written from within the TSE via the Configuration Access Correlation Trusted Capability.	RO

Add Sections 7.21, 7.21.1, 7.21.2, 7.21.2.1, 7.21.2.2, 7.21.3, 7.21.3.1, 7.21.3.2, and 7.21.3.3 as follows:

## 7.21 Trusted Configuration Space Header

Figure 7-90 details allocation for register fields of the Trusted Configuration Space header for PCI Express Trusted Devices; Table 7-77 provides the respective bit definitions. The Trusted Configuration Space header identifies the device class to software running in the Trusted Software Environment, and it points to the first Trusted Capability structure.



**Figure 7-90: Trusted Configuration Space Header**

**Table 7-77: Trusted Configuration Registers**

<b>Offset (bytes)</b>	<b>Size (bits)</b>	<b>Description</b>	<b>Register Attributes</b>
<u>000h</u>	<u>24</u>	<u><b>Trusted Class Code</b> – This is formatted as three 8 bit fields (Base Class, Sub-class and Programming Interface), structured identically to the class code in PCI 3.0 Type 0 and Type 1 Configuration Space headers. The Trusted Class Code refers to the trusted functionality provided by the device, and this field is permitted to be different than the class code field in Standard Configuration Space. These IDs are allocated by the PCI SIG from the same pool as conventional PCI class codes.</u>	<u>RO</u>
<u>004h</u>	<u>12</u>	<u><b>First Trusted Capability Offset</b> – This field contains the offset to the first PCI Express Trusted Capability structure. This offset is relative to the beginning of the device's Trusted Configuration Space.</u>  <u>The bottom two bits of this offset are reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.</u>	<u>RO</u>

## **7.21.1 Trusted Capabilities Header**

575 Trusted Capabilities are located in Trusted Configuration Space at offsets higher than the end of the TCS Header. The TCS Header includes a pointer to the first Trusted Capability structure to allow for future expansion of the header.

580 PCI Express Trusted Capability structures are allocated using a linked list of optional or required PCI Express Trusted Capabilities following a format resembling PCI Express Extended Capability structures. The first DWORD of the capability structure identifies the capability/version and points to the next capability. Each Trusted Capability structure must be DWORD aligned.

Trusted Capability IDs are allocated by the PCI-SIG from a separate pool than that used for PCI Express Extended Capability IDs.

585 It is recommended that all registers exposed in Trusted Configuration Space be contained within Trusted Capability structures. (The exception to this is the TCS Header itself, which precedes the Trusted Capability list.) The Vendor-Specific Trusted Capability (VSTC), defined in Section 7.21.3, provides a mechanism for vendors to include registers in TCS in a standard, discoverable manner.

## **7.21.2 Configuration Access Correlation Trusted Capability**

590 The PCI Express Configuration Access Correlation (CAC) trusted capability is a required trusted capability for any PCI Express device that provides a Trusted Configuration Space. The CAC is a read/write 32-bit value that is used by software running within the TSE to validate that Standard Configuration Space accesses performed via ECAM are targeting the same device as Trusted Configuration Space accesses performed via the corresponding TCAM.

595 PCI Express devices that implement this Trusted Capability must also implement the corresponding CAC extended capability in Standard Configuration Space (see Section 7.20). Both capabilities must



access the same register for the Device Correlation—it is by this means that correlation is verified by software running in the TSE.

Figure 7-91 shows the PCI Express Configuration Access Correlation Trusted Capability structure.

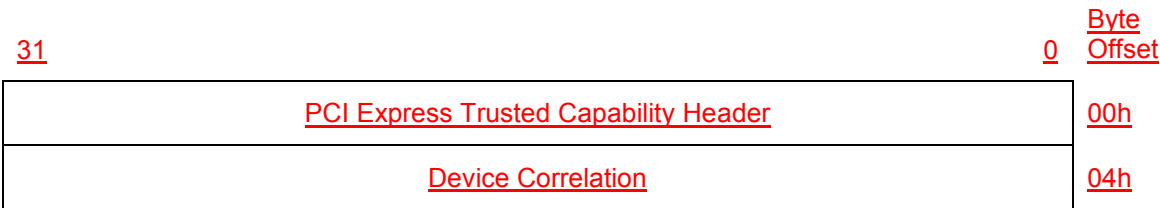
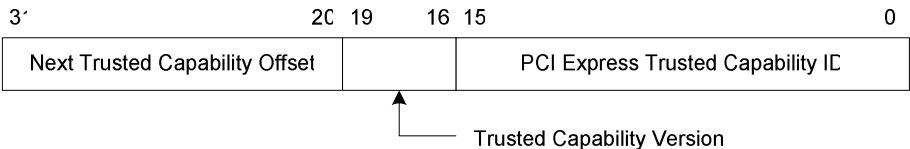


Figure 7-91: Configuration Access Correlation Trusted Capability

600

**7.21.2.1 Configuration Access Correlation Trusted Capability Header (Offset 00h)**

Figure 7-92 details allocation of register fields in the CAC Trusted Capability header; Table 7-78 provides the respective bit definitions. The Trusted Capability ID for the CAC Trusted Capability is 0001h.



605

Figure 7-92: Configuration Access Correlation Trusted Capability Header

Table 7-78: Configuration Access Correlation Trusted Capability Header

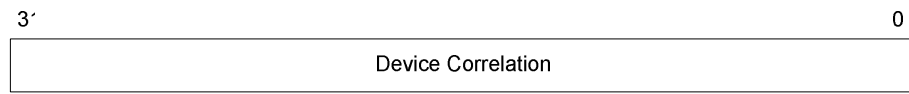
Bit Location	Register Description	Attributes
15:0	<b>PCI Express Trusted Capability ID</b> – This field is a PCI-SIG defined ID number that indicates the nature and format of the trusted capability.  Trusted Capability ID for the CAC Trusted Capability is 0001h.	RO
19:16	<b>Trusted Capability Version</b> – This field is a PCI-SIG defined version number that indicates the version of the capability structure present.  Must be 1h for this version of the specification.	RO
31:20	<b>Next Trusted Capability Offset</b> – This field contains the offset to the next PCI Express Trusted Capability structure or 000h if no other items exist in the linked list of capabilities. This offset is relative to the beginning of the device's Trusted Configuration Space.  The bottom two bits of this offset are reserved and must be implemented as 00b although software must mask them to	RO

allow for future uses of these bits.

### 7.21.2.2 Device Correlation Register (Offset 04h)

The Device Correlation register is a 32-bit read/write register that is used by software running within the TSE to confirm that Standard Configuration Requests and Trusted Configuration Requests for a given bus, device, and function are targeting the same device.

Figure 7-93 details the allocation of register fields for the Device Correlation register. Table 7-79 provides the respective bit definitions.



**Figure 7-93: Device Correlation Register**

**Table 7-79: Device Correlation Register**

Bit Location	Register Description	Attributes
31:0	<b>Device Correlation</b> – This field contains a value, written from within the TSE, that is read back in Standard Configuration Space via the Configuration Access Correlation Capability.	RW

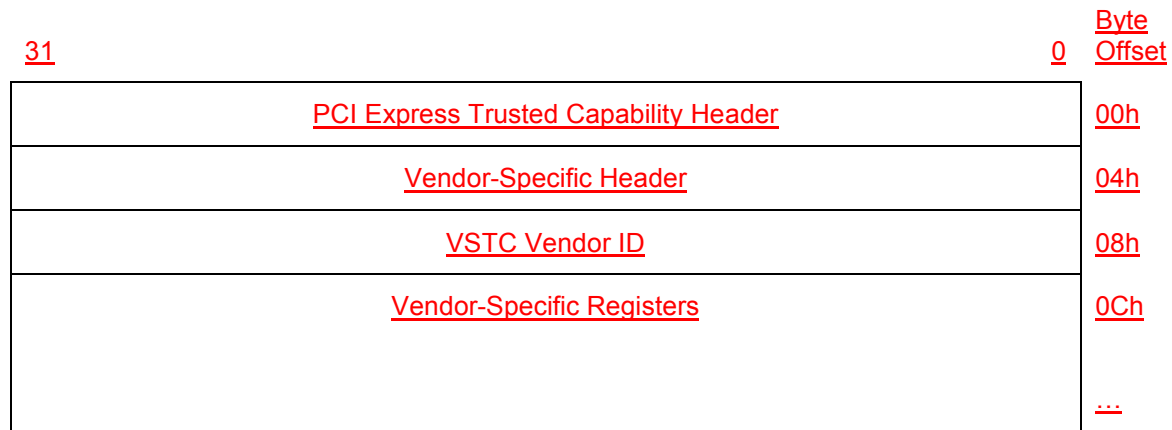
### 7.21.3 Vendor-Specific Trusted Capability

The PCI Express Vendor-Specific Trusted Capability (VSTC) is an optional trusted capability that is permitted to be implemented by any PCI Express device that provides a Trusted Configuration Space. This allows members of the PCI SIG who have an assigned vendor ID to define vendor-specific trusted capabilities which can then be exposed by PCI Express Trusted Devices via TCS.

A single PCI Express Function is permitted to contain multiple VSTC structures.

The VSTC in Trusted Configuration Space serves a similar purpose to that of the Vendor-Specific Extended Capability (VSEC) in Standard Configuration Space, as described in Section 7.18. The VSTC differs from the VSEC in that PCI Express functions are permitted to supply (public) VSTCs defined by vendors other than the device's vendor (as identified by the Vendor ID register located at byte offset 00h in PCI-compatible Configuration Space).

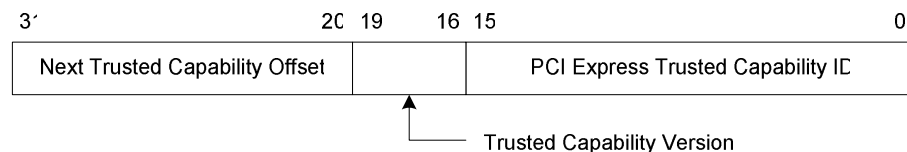
Figure 7-94 details allocation of register fields in the VSTC structure. The structure of the PCI Express Trusted Capability Header, the Vendor-Specific Header, and the VSTC Vendor ID is architected by this specification. The structure and definition of the Vendor-Specific Registers area is determined by the vendor indicated in the VSTC Vendor ID register located at offset 08h within the VSTC structure.



**Figure 7-94: PCI Express VSTC Structure**

### 7.21.3.1 Vendor-Specific Trusted Capability Header (Offset 00h)

Figure 7-95 details allocation of register fields in the Vendor-Specific Trusted Capability header; Table 7-80 provides the respective bit definitions. The Trusted Capability ID for the Vendor-Specific Trusted Capability is 0002h.



**Figure 7-95: Vendor-Specific Trusted Capability Header**

**Table 7-80: Vendor-Specific Trusted Capability Header**

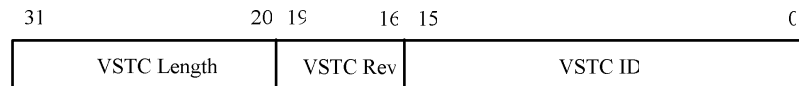
Bit Location	Register Description	Attributes
15:0	<b>PCI Express Trusted Capability ID</b> – This field is a PCI-SIG defined ID number that indicates the nature and format of the trusted capability.  Trusted Capability ID for the Vendor-Specific Trusted Capability is 0002h.	RO
19:16	<b>Trusted Capability Version</b> – This field is a PCI-SIG defined version number that indicates the version of the capability structure present.  Must be 1h for this version of the specification.	RO
31:20	<b>Next Trusted Capability Offset</b> – This field contains the offset to the next PCI Express Trusted Capability structure or 000h if no other items exist in the linked list of capabilities. This offset is relative to the beginning of the device's Trusted Configuration Space.  The bottom two bits of this offset are reserved and must be implemented as 00b although software must mask them to	RO

allow for future uses of these bits.

### 7.21.3.2 Vendor-Specific Header (Offset 04h)

Figure 7-96 details allocation of register fields in the Vendor-Specific header; Table 7-81 provides the respective bit definitions.

Vendor-specific software must qualify the VSTC Vendor ID (located at offset 08h, detailed in Section 7.21.3.3 below) before attempting to interpret the values in the VSTC ID or VSTC Rev fields.



**Figure 7-96: Vendor-Specific Header**

**Table 7-81: Vendor-Specific Header**

Bit Location	Register Description	Attributes
15:0	<b>VSTC ID</b> – This field is a vendor-defined ID number that indicates the nature and format of the VSTC structure.  VSTC IDs are divided into two groups: public and private. Public VSTC IDs must set bit 15 to 1b and private VSTC IDs must set bit 15 to 0b. VSTCs with private VSTC IDs must only be used by PCI Express functions where the VSTC Vendor ID matches the Vendor ID presented in the PCI 3.0 Type 0 or Type 1 configuration space header for the Trusted Device. The use of VSTC structures with public VSTC IDs is permitted for PCI Express functions where the Vendor ID presented in the PCI 3.0 Type 0 or Type 1 configuration space header does not match the VSTC Vendor ID <sup>4</sup>	RO
19:16	<b>VSTC Rev</b> – This field is a vendor-defined version number that indicates the version of the VSTC structure.  Software must qualify the VSTC Vendor ID and VSTC ID before interpreting this field.	RO
31:20	<b>VSTC Length</b> – This field indicates the number of bytes in the entire VSTC structure, including the PCI Express Trusted Capability Header, the Vendor-Specific Header, the VSTC Vendor ID, and the Vendor-Specific Registers.	RO

<sup>4</sup> Any restrictions on the use of VSTCs with public VSTC IDs are beyond the scope of this specification. Further, the disclosure of public VSTC IDs is solely at the discretion of the individual Member of PCI-SIG who makes such disclosure. In making such disclosure, the Member waives any and all claims against PCI-SIG with regard to third party use of its VSTC ID.

### 7.21.3.3 VSTC Vendor ID (Offset 08h)

Figure 7-97 details allocation of register fields in the VSTC Vendor ID; Table 7-82 provides the respective bit definitions.

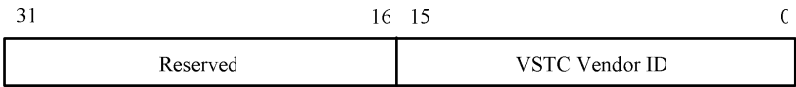


Figure 7-97: VSTC Vendor ID

Table 7-82: VSTC Vendor ID

Bit Location	Register Description	Attributes
15:0	<p><b>VSTC Vendor ID</b> – This field is the Vendor ID, as defined by PCI-SIG, of the vendor defining this Vendor-Specific Trusted Capability. If the associated VSTC ID is public (see discussion of VSTC ID in Table 7-81), then the VSTC Vendor ID is permitted to be different than the Vendor ID presented in the PCI 3.0 Type 0 or Type 1 configuration space header for the Trusted Device. Otherwise, the VSTC Vendor ID must be the same as the Vendor ID presented in the PCI 3.0 Type 0 or Type 1 configuration space header for the Trusted Device.</p> <p>Software must qualify the VSTC Vendor ID before interpreting the VSTC Rev and VSTC ID fields in the Vendor-Specific Header (offset 04h).</p>	RO
31:16	<b>Reserved</b> – Must hardwire the field to 0000h.	RsvdP

## **X. Trusted Configuration Space Platform Implications** **Appendix**

### **X.1. Trusted Software Environment Requirements**

The introduction of PCI Express Trusted Configuration Space enables the design of trusted computing platforms capable of providing compliant devices with assurance that their Trusted Configuration Registers can only be accessed by software running in a Trusted Software Environment (TSE). Establishing device trust in the software that initiates Trusted Configuration Requests makes it possible to secure and control access to certain secret, sensitive, or personally-identifiable information these devices may contain (e.g., a uniquely-identifying public key or certificate which the device provides for revocation purposes). While a full description of a Trusted Software Environment is platform-specific and beyond the scope of this specification, the following requirements apply to any system implementing a TSE capable of issuing Trusted Configuration Transactions over PCI Express:

**1. The platform must be able to restrict which software is allowed to access the Trusted Configuration Access Mechanism (TCAM). The level of effort involved in enforcing this restriction is dependent upon how “field configurable” the software environment is:**

- a. For some embedded system applications, it may be sufficient to simply lock down the software configuration such that no unauthorized software can be added to or run on that system.**
- b. In more “flexible” environments (e.g., a PC running a general-purpose OS) hardware support at the processor and supporting chipset level is required in order to identify and distinguish between software running within a TSE and software running outside of a TSE, and to only allow TCAM access by software running inside of a TSE.**

**2. For the more flexible environments (i.e., 1.b above), some owner-accessible mechanism must be provided to allow for assignment of one or both of the following capabilities to a given piece of software:**

- a. To run in the TSE**
- b. To access the TCAM when running in the TSE**

**This explicit “opt-in” model provides assurance to the Trusted Device that any Trusted Configuration Request it receives originated from software authorized by the machine owner to make that request.**

**3. If a platform allows software outside of the TSE to run concurrently with software inside the TSE, then the platform must provide a mechanism for software running in the TSE to monitor and/or control configuration changes (e.g., bus number assignment) that occur during the time TCAM is being used.**

**4. The platform must prevent software running outside of the TSE (including firmware) from virtualizing or emulating devices accessed via the TCAM.**

**5. If a platform supports PCI Express add-in cards, the platform must ensure that those devices cannot perform DMA to the TCAM Region.**

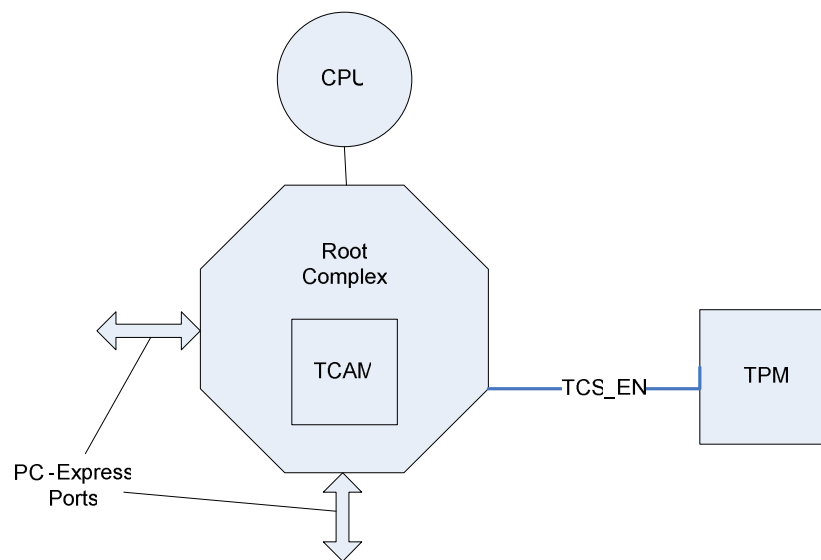
### **X.2. Example TCS/TSE Implementation**

The Trusted Computing Group (TCG) has produced a specification for a trusted subsystem that can be used in platforms containing the necessary enabling hardware. (See <http://www.trustedcomputinggroup.org> for more details.) One important hardware component required by that specification is the Trusted Platform Module (TPM).

The TPM can optionally provide a number of general purpose input/output (GPIO) pins. Access to and control of these pins can be gated by the TPM's rich access control semantics which include the ability to specify what software must be running in the Trusted Software Environment. TPM GPIO pins are in an asserted state when they are at low voltage.

In this example, the root complex for a platform compliant with the v1.2 TCG specification provides a TCS\_EN# pin as a mechanism to enable Trusted Configuration Space accesses. When the TCS\_EN# pin is deasserted (high), all accesses made to Trusted Configuration Space via the TCAM are ignored. Reads return all 1 bits.

The system board for our example implementation would connect TPM GPIO pin 0 to the TCS\_EN# pin of the root complex as illustrated in Figure X-1 below.



**Figure X-1: Gating TCS Access Via TPM**

When software running in the TSE wishes to access Trusted Configuration Space, it would first issue a TPM GPIO command to cause the TPM to assert TCS\_EN# (drive it low) in order to enable the mechanism. A v1.2 TPM can differentiate requests, and in particular TPM GPIO command requests, that are sent by software running in the TSE, using a combination of TPM hardware and firmware features. Using these features, the TPM would give access to GPIO pin 0 to only the software that has been authorized to access Trusted Configuration Space and would restrict the ability of unauthorized software to initiate TCS requests.